DOCUMENT RESUME

ED 391 482 IR 017 635

AUTHOR Jones, Marshall G.
TITLE Anthropomorphizing the User Interface: A Case for Interface Guides.
PUB DATE [95]
NOTE 9p.; In: Eyes on the Future: Converging Images, Ideas, and Instruction. Selected Readings from the Annual Conference of the International Visual Literacy Association (27th, Chicago, IL, October 18-22, 1995); see IR 017 629. Contains figures which may not reproduce clearly.
PUB TYPE Speeches/Conference Papers (150) -- Viewpoints (Opinion/Position Papers, Essays, etc.) (120)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Computer Interfaces; *Computer Literacy; *Computer Software Development; Instructional Effectiveness; Multimedia Materials; User Needs (Information); *Visual Aids
IDENTIFIERS *Anthropomorphism; Beginning Competence; Clip Art; Iconic Representation; *Navigation (Information Systems)

ABSTRACT

Anthropomorphism can be defined as the attribution of human characteristics or behavior to inanimate objects. People give names to their automobiles and computers as a way to relate to complicated pieces of technology that they use regularly, but do not fully understand. Software designers may claim that their interfaces are intuitive, but in fact, software is too contrived to be intuitive. If navigating the program seems easier it is largely because the user has become more accustomed to programs of that type. Users have difficulty arriving at that point, however, without some built-in help, like interface guides. Interface guides, or agents, are navigational devices in the user interface which take action on behalf of the user. Some agents have been named and anthropomorphized into figures like a butler, a miner, or a wizard, and they can set forth the user's schedule, guide him through a program or online information resource, and offer feedback and encouragement. Interface guides can personalize and customize the interface, add structure to the software by organizing content around a specific guide, and integrate motivation and navigation. With video clips and snippets of audio, they do require a lot of "zorch," or processing power, as well as a lot of disk space, but many users already have both of these things in abundance. The major problems lie in the fact that more research on their effectiveness is needed and in the fact that much of the clip art that some designers depend on is rife with gender and racial stereotypes. Four reproductions of computer screens illustrate the discussion. (Contains 13 references.) (BEW)

Anthropomorphizing The User Interface: A Case For Interface Guides

Marshall G. Jones

Anthropomorphism and Interaction

Anthropomorphism, as defined by The American Heritage Dictionary, third edition, is the "attribution of human motivation, characteristics, or behavior to inanimate objects, animals, or natural phenomena." Anthropomorphizing inanimate objects is something that we do on a daily basis. People name their cars, boats, and computers. A colleague of mine told me that he was having problems transferring a file to me via the internet. His comment was, "Calvin (the name of his computer) must be feeling a little off today." Our computers are out to get us; our cars are feeling good. Everyday we attribute human characteristics to non-living objects. It is a technique we use to help our human selves relate to complicated pieces of technology that we use regularly, but do not fully understand.

Computer software has long attempted to make its complex algorithms accessible to the user. Software designers and developers continue to strive to make their interfaces *intuitive*. I argue that software, by its very nature, cannot be intuitive. It is a contrived environment. It is not like anything else. For people to be able to intuit what to do next, they must have a frame of reference to begin from. While it is certainly true that because I can use one word processor, I can use any word processor, it is not because the new word processor is inherently intuitive. It is because somewhere in my past, I learned to use a word processor. The functions of a word processor were taught to me, either by someone else, or by a manual or other piece of instruction. I can intuit what needs to be done only because I have a frame of reference based in my experience with using application software, in this case, a word processor. Educational software is even more problematic.

In educational software, people have the luxury of repeated use. While my word processor may offer thousands of functions, I may only use one hundred of them on a regular basis. When faced with a complex task, I can work through the problem because I have experience in using the word processor. Once again, through the frame of reference of my experience, I can learn new functions, and perform more complex tasks because of my repeated, daily use, of my word processor. In educational software,

people do not have the luxury of repeated use. In many cases, we expect our software to be used once, maybe two or three times. Users are in the software to gain the knowledge and or skills they need, and then they leave the software (Jones, 1995). Consequently, they typically do not develop the skills they need to perform complex tasks in the software. The software needs to be obvious, sometimes painfully so. When designers and developers watch people use their software, they are often amazed that people do not know what to do, or that they miss important features of the software. What is obvious to us, because of our experience in using, designing, and developing educational software, is not always obvious to novice users. Schneiderman (1987) says that software needs to have progressive help. Features should be built in to the software that make it easy for a novice to understand, but that shortcuts should be built in to allow for the development of *power users*. This is quite evident in application software. In nearly every word processor available today, the user can give the commands to copy and paste by selecting the options from the **Edit** menu. However, most people, as they become more experienced in word processing, will bypass the menu in favor of the keyboard equivalent (e.g. command/control C for copy, command/control V for paste.) Be it educational software or application software, users need tools available to them to help them begin the often times complex task of learning to use their programs.

Microsoft products help soften the blow of entering sophisticated pieces of application software by providing us with *Did You Know* statements when we open the software. Computer games provide us with dramatic representations of *real life* to help us understand and interact with them. The purpose is to take a complicated activity and make it seem some how non threatening. Or, to use the well worn cliché, user friendly.

Anthropomorphism in the computer interface is not only about making our programs look life like, but it is also about making us think that our programs *know* us. One example of this is the oft used technique of using a person's name in a piece of instructional software (Alessi & Trollip, 1991; Keller & Suzuki, 1988). When the user starts the program, their name is entered. As they make progress, their name is used in dialogue boxes telling them of their triumphs and failures, thus attempting to give the user the impression that the computer knows them, and is somehow on their side.

Early computer-based instruction (CBI) was fairly linear. It was hierarchical in nature, and users found themselves working through menu items to complete different sections. Navigation was simple. You progressed from one screen to another by pressing the space bar, or clicking on the right arrow, to continue. Jumps between ideas, when they were made, were typically made back to the main or previous menu, where another section was selected.

In today's complex multi-media software, users are faced with a daunting task. Large learning environments are difficult to manage. Sophisticated environments which combine multi-media and hypertext capabilities provide users with a wealth of options (Jones, Farquhar, & Surry, 1995). Jonassen, (1988) was one of the first to recognize that allowing users to work in an associative manner, i.e. allowing users to follow their own path rather than the path set forth by the designer, could help deepen the level of processing of the information by the learner. This is good. But to make those links, the software must provide for those links, which means that the designer must discern them, build them in, and make access available to them by the learner. This is hard. And it is not hard just for the designer. It is hard for the user as well. Well ordered hierarchical systems are organized fairly succinctly by the use of menus. Interactive learning environments are often difficult to

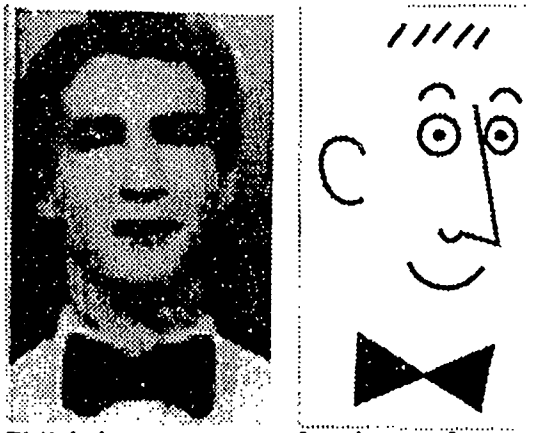
represent with a traditional menu. Different methods of navigation are in place; comparable methods of organization are not.

Interface Agents and Interface Guides

Interface guides may be subsumed by a concept known as Interface Agents (Laurel, 1990; Laurel, Oren, & Don, 1992). Interface agents are navigational devices in the user interface which take action on behalf of the user. They do something for the user. These have been demonstrated in commercials by AT&T, where an intelligent agent, represented as a happy, tail wagging dog, tells a groggy, coffee drinking woman what is in store for her during the day, including his attempts at trying to get her playoff tickets.

Apple Computer also theorized about the possibilities of an interface agent in their futuristic promotional videotapes (Laurel, 1990). "Phil" is an electronic butler of sorts who manages the affairs of a college professor (See Figure 1).

Figure 1
Two representations of "Phil." The bow tie becomes an iconographic symbol, making the graphic recognizable in either representation. (From Laurel, 1990, page 365)



Phil brings up maps for the professor, contacts colleagues who will do guest lectures, and reminds him of his mother's birthday. The purpose of the agent is to manage the complex internal capabilities of the computer, freeing up the user to do what the user would rather do. Voice recognition software allows the user to work with the computer without being in front of the computer. But it does more than that: it makes the computer human in ways that we can, at the moment, only imagine, but can still understand. This concept of the interface agent, like most literature on user interface design, focuses on system software and application software. The question becomes how can this idea be extrapolated to become relevant to educational software?

Interface Guides in Educational Software

Oren, Saloman, Kreitman, and Don (1990) attempt to answer this question by introducing the idea of an interface guide. Interface guides are navigational and organizational emissaries that exist within the educational software. They represent the best interests of the user. A guide may become an internal coach for the user, suggesting direction, providing feedback, and even encouragement, as the user works within a large learning environment.

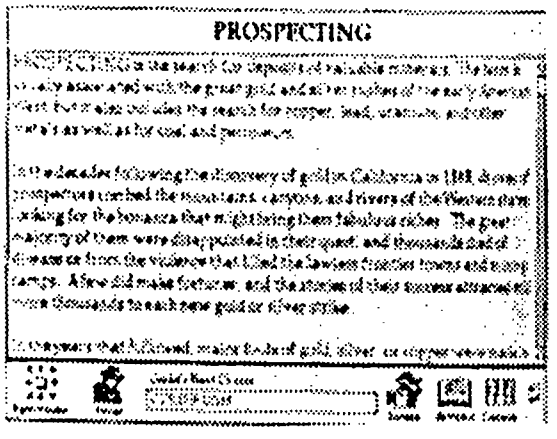
Examples of interface guides in software

Interface guides are probably best described within the context of one of the earliest CD-ROM encyclopedias Oren, Saloman, Kreitman, & Don (1990). A program was being prototyped that would serve as an educational database covering the American history from 1800—1850. To help users navigate the large amounts of information contained in the program, a metaphor of an interface guide was used. These guides represented people who would have lived during 1800—1850. Depending on their interest, the user could select one of the interface guides, (See Figures 2 &3). Clicking on the miner guide would return information in the database relative and relevant to the

life of a miner. The same would be true for any of the other guides.

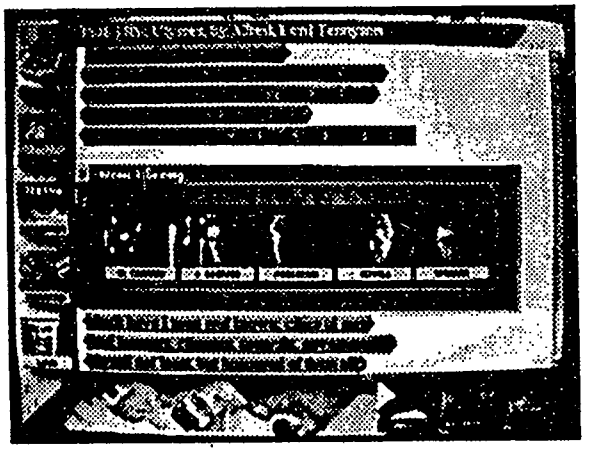
Figure 2
The list of guides from Oren, Saloman, Kreitman, & Dor (1990)

Figure 3
Information returned by guide, Miner



IBM's groundbreaking multi-media extravaganza, *The Illuminated Books and Manuscripts* provide another example of how interface guides may be used. In the exploration of Tennyson's *Ulysses*, users have the option of having the poem, or sections of the poem, read to them in different voices represented by different actors. From Danny Glover to Rod Steiger, users have a wide range of people to help them understand the poem through the use of dramatic readings (See Figure 4). These guides, though not explicitly referred to as guides in the software, provide the user with pedagogical help rather than navigational help. It also adds a bit of the human touch to the program by bringing the work of a dead poet to life by having it read by contemporary figures.

Figure 4
Guides used to explicate content from IBM's *Illuminated Books and Manuscripts.*



The Theoretical Foundations of Interface Guides

Keller and Suzuki (1988) begin, however tacitly, to introduce the notion of interface guides through the use of motivational strategies. They suggest that one way to provide motivation is through the use of human interest language and graphics. In describing the wizard in a 1985 program *Carrying*, the authors speak of the use of the person in the program serving as a surrogate teacher for the users of the program. Even in the seemingly ancient graphic display capabilities of the 1985 program, Keller and Suzuki point out that facial expressions of the wizard change on right and wrong answers to provide feedback. But Keller and Suzuki are victims of time. While they discuss the wizard, the graphics they are using seem obviously dated. They are; don't be fooled. Conceptually they are describing something that was not yet technically possible. Today we have the luxury of morphing software, digitized movies,

high resolution display devices, and monitors capable of displaying millions of colors—the very tools we need to display more sophisticated guides. But many of the theories suggesting the use of guides exist in Keller's ARCS model, and Jonassen's (1988) early discussions of generative learning environments. The technology available to us today provides us with the storage space needed to include digitized video and audio in our program. Many desktop machines today have more Random Access Memory (RAM) than the machines of 10 years ago had storage space. This provides us with enormous opportunities. However, we would be ill advised indeed to go about this simply because we can. Rather, guides should be used to provide navigational, structural, and pedagogical aid to the user.

Menus in early pieces of instructional software provided the navigation and structure that programs needed. Motivational elements were often added as an after thought. Users studied; answered questions, and were then treated to animations or sounds on right answers. Interface guides provide us with the means to provide organization, navigational help, and motivation within the software itself. Keller and Suzuki suggest that it is preferable to integrate motivational elements into the software seamlessly. But the reality was that, at the time of the writing, that was difficult to do. Jones, Farquhar, and Surry (1995) tell us that the interface of the program is a cohesive whole representing the content in the program, the visual representation of the program, and the control of the program. Interface guides provide us with a possible means for achieving the end of motivational strategies integrated into the navigational and pedagogical structure of the program.

Issues in the use of interface guides

Interface guides may provide a means of making the software easier to use. Oren, Saloman, Kreitman, and Don (1990) state that the reaction to the use of interface guides in their program was quite favorable. There are few programs on the market which employ interface guides. Consequently, most of us know very little about using an interface guide. The following is a series of issues to consider when designing and implementing interface guides.

1. Personalize and customize the interface.

While we have been told many times that personalizing the software is a good thing to do (e.g., Alessi & Trollip, 1991; Keller & Suzuki, 1988), interface guides offer not only personalization, but customization as well. A particular guide may provide the learner with exactly the direction they need. In the example by Oren, Saloman, Kreitman, and Don (1990), the ten guides offered provide a remarkable amount of flexibility. Not total flexibility, surely, but more than simply following the menu structure. Flexibility here refers to the program's ability to allow users different representation of the same content relative to the guide they select.

2. Add structure to the software by organizing content around a specific guide.

As stated earlier, menus do this as well. An individual guide can help users structure the program around their area of interest. The guides can also help users manage the information in the program by serving as signposts in the program (Jones & Okey, 1995) Once a guide is selected, a representation of the guide may be placed on the screen as a visual reminder of what the user is looking at and the perspective on the content that the particular guide brings to the program (See Figure 3).

3. **Integrate motivation and navigation.**

In many programs, navigation is a value added feature. After the content has been designed and developed, people then consider how their users should have access to the information. Using interface guides would demand that design and development issues be considered simultaneously. All information related or provided by the guides will have to be created. The system messages you will provide to users will need to be scripted. If you are digitizing your guides, the video will need to be shot. You will need to attend to these details as rigorously as you attend to your content.

4. **Zorch.**

Because of the tremendous power available to us in our computers today, we have the ability to create more creative, interesting, and realistic interactions. Saloway, (1995) tells us that our computers have more "zorch" than they did ten years ago. Zorch refers to the amount of processing power a computer has. In the early days of the microcomputers, all of the computer's zorch went to the application itself. There was not enough power to make the interface easier to use. Today, we have the processors to make the programs not only powerful, but easier to use. Using part of your computer's zorch for an interface guide, or a series of interface guides, may make the program easier for users to understand.

5. **Uncharted territory.**

While Oren, Salomon, Kreitman, and Don (1990) tell us that reaction to the use of interface guides is favorable, we don't know much about their effectiveness, nor the best way to implement them. More exploration is needed in using interface guides. More research is needed in regards to their effectiveness.

6. **System hogs.**

If you decide to use digitized video and audio to make your guides come alive, then you will have a problem of space and system resources. A single 10 second video clip with audio can require 500k to 1 megabyte of storage space. In a large program where multiple guides are used, you could be talking about needing 40 to 80 megabytes of space simply for your guides. This is a lot of space, no doubt. However, CD-ROMs provide us with storage options, and the simple fact of the matter is that the gigabyte hard drive is no longer a fantasy but a reality. While much space is needed, the fact is that we may already have it.

7. **What do guides look like?**

You do not need to rely on video and audio for your guides. Static graphics and text messages take up less space, and may be as useful as video based guides. However, most of us, particularly those of us who develop on shoestring budgets, rely on clip art. And there are problems with clip art. Binns and Branch (1995) report that much of the clip art that we have available to us is rife with gender and racial stereotypes. Care must be taken in choosing your guides. Whether they are video with audio, or static graphics with text, you must decide on what your guides will look like, what they will say, and how they will say it. Guides are intended to be advocates of the user who exist within the software. An advocate should be somebody you can trust and relate to. To have all of your guides look the same, i.e. being all Caucasian, all male, all female, or all African American, may not be best for your users. Users should have the freedom to pick the person they want to represent their best interests

in the software. Front end analysis needs to be done in the development of any learning environment. We are quite familiar with using front end analysis to analyze our audiences. A careful analysis of the audience will help to determine what a guide should look like. An analysis of the instructional environment will tell us what system resources are available to us, thus helping us to determine whether the guides should be digitized video and audio or static with text.

Conclusion

Interface guides may be a viable option in the development of the interaction style of a computer based learning environment. They can provide personalized and customized programs for multiple users of learning environments. By providing even greater human characteristics to a piece of software, they can anthropomorphize the user interface, thus making the software more relevant to the users. Large computer based learning environments, though engaging, can be harrowing places for a user to be. Guides can help lead them through the environment successfully.

References

Alessi, S. & Trollip, S. (1991). *Computer-based instruction: Methods and development.* Englewood Cliffs, NJ: Prentice-Hall.

Binns, J. C. & Branch, R. C. (1995). Gender stereotyped computer clip-art images as an implicit influence in instructional message design. In D. G. Beauchamp, R. A. Braden & R. E. Griffin (Eds.), *Imagery and Visual Literacy.* (pp. 315-324). The International Visual Literacy Association.

Jones, M. G. (1995). Visuals for information access: A new philosophy for screen and interface design. In D. G. Beauchamp, R. A. Braden & R. E. Griffin (Eds.), *Imagery and Visual Literacy.* (pp. 264-272). The International Visual Literacy Association.

Jones, M. G., Farquhar, J. D. & Surry, D. W. (1995). Using metacognitive theories to design user interfaces for Computer-based learning. *Educational Technology* 35(4) pp. 12-22.

Jones, M. G. & Okey, J. R. (1995). Interface design for computer-based learning environments. This paper has been published by *Instructional Technology Research Online (InTRO)* at the WWW site: **[http://129.7.160.78/InTRO.html]** on February 21, 1995.

Jonassen, D. H. (1988). Integrating learning strategies into courseware to facilitate deeper processing. In D. Jonassen (Ed.), *Instructional designs for microcomputer courseware.* (pp. 151-181). Hillsdale, NJ: Lawrence Erlbaum Associates.

Keller, J. M. & Suzuki, K. (1988). Use of the ARCS motivation model in courseware design. In D. Jonassen (Ed.), *Instructional designs for microcomputer courseware.* (pp. 401-434). Hillsdale, NJ: Lawrence Erlbaum Associates.

Laurel, B. (Ed.). (1990). *The art of human-computer interface design.* Menlo Park, CA: Addison Wesley.

Laurel, B., Oren, T., & Don, A. (1992). Issues in multimedia design: Media integration and interface agents. In M. M. Blattner & R. B. Dannenberg (Eds.), *Multimedia interface design.* (pp. 53-64). ACM Press.

Laurel, B. (1990). Interface agents: Metaphors with character. In B. Laurel (Ed.), *The art of human computer interface design.* (pp. 355-366). Menlo-Park, CA: Addison Wesley Publishing Company, Inc.

Oren, T., Saloman, G., Kreitman, K, & Don, A. (1990). Guides: Characterizing the user interface. In B. Laurel (Ed.), *The art of human computer interface design.* (pp. 367-382). Menlo-Park, CA: Addison Wesley Publishing Company, Inc.

Saloway, E. (1995). Keynote address at the annual conference of the Association for Educational Communications and Technology. Anaheim, California. February, 1995.

Schneiderman, B. (1987). *Designing the user interface: Strategies for effective human-computer interaction.* Menlo Park, CA: Addison-Wesley.